Методичка 3.2 - Процесс загрузки PE-файлов

Общая структура PE-файла

Программа prog-1.4.3.exe использует 2 внешние функции MessageBox из библиотеки user32.dll и ExitProcess из kernel32.dll. Когда программа использует функции из внешних библиотек, этот процесс называется импортом. Говорят, что PE-файл импортирует некоторые функции из динамических библиотек.

Для того, чтобы программы могла использовать функции из динамических библиотек их необходимо подгрузить в то же адресное пространство процесса, где работает сама программа. Копирование программы в память, а также подключение всех необходимых динамических библиотек выполняет загрузчик PE-файлов.

Загрузчик PE-файлов как-то должен понять, какие библиотеки необходимо подгрузить при загрузке программы. Должно быть место, где указана эта информация. И в PE-файле есть такое место. Называется оно - таблица импорта. Из этой таблицы PE-загрузчик получает список необходимых библиотек и функций. А динамические библиотеки имеет таблицу экспорта, по которой PE-загрузчик ищет RVA нужной функции.

Алгоритм вычисления RVA таблицы импорта

DATA_DIRECTORY
...
    > *DIRECTORY_ENTRY_IMPORT*
    Name RVA
    Name
    OriginalFirstThunk
    …
    FirstThunk
    …
  > *DIRECTORY_ENTRY_IAT*
    struct IMPORT_ADDRESS_TABLE

Для того, чтобы узнать точное расположение таблицы импорта в PE-файле, необходимо обратиться к полю DATA_DIRECTORY, которое находится в опциональном заголовке (Optional Header).

Это поле называется директорией данных!

Директория данных содержит RVA и размеры важных структур данных PE-файла. Таблица импорта как раз одна из таких структур. Структуры данных расположены друг за другом. Размер каждого блока 8 байт (RVA - 4 байта и размер - 4 байта). Таблица импорта идет второй, значит, чтобы получить RVA на таблицу импорта необходимо к RVA на DATA_DIRECTORY прибавить 8.

Получается, алгоритм вычисления RVA таблицы импорта состоит из двух шагов:

1. Из опционального заголовка (Optional Header) происходит получение RVA директории данных.
2. Рассчитывается смещение внутри директории данных до таблицы импорта (DIRECTORY_ENTRY_IMPORT) по формуле:

   RVA директории данных + (позиция нужной структуры * размер структуры)

   Для таблицы импорта:

   RVA директории данных + (1 * 8) = RVA директории данных + 8

Структура таблицы импорта

```
DIRECTORY_ENTRY_IMPORT
	struct IMPORT_DESCRIPTOR
	{
		OriginalFirstThunk		// Import Lookup Table RVA
		Time/Date Stamp
		Forwarder Chain
		Name RVA			// например, kernel32.dll
		FirstThunk			// Import Address Table RVA
	}
```

Таблица DIRECTORY_ENTRY_IMPORT, она же таблица импорта, содержит записи по каждой библиотеке, которая должна быть загружена. Каждая запись хранится в структуре IMPORT_DESCRIPTOR. Структуры расположены друг за другом, то есть это массив расположенных подряд структур. Их количество равно количеству библиотек, которые необходимо подключить.

OriginalFirstThunk указывает на lookup-таблицу (Import Lookup Table), содержащую имена импортируемых функций.

FirstThunk указывает на таблицу адресов импортируемых функций (Import Address Table). Сюда PE-загрузчик будет писать актуальные адреса функций, после того, как он их узнает.

Name RVA - это RVA строки, содержащей имя библиотеки, из которой будет производиться импорт.

Описание структуры IMAGE_IMPORT_BY_NAME

```
struct IMAGE_IMPORT_BY_NAME
{
```

Hint
Name1 // например, ExitProcess
}

Через OriginalFirstThunk происходит получение RVA на массив структур IMAGE_IMPORT_BY_NAME, которые содержат имена функций.

```
struct IMAGE_IMPORT_BY_NAME
{
    Hint
    Name1 // например, ExitProcess
}
```

PE-загрузчик проходит по всему массиву и через структуру IMAGE_IMPORT_BY_NAME получает имена функций. После того, как он получает имя библиотеки и имена функций он может определить адреса этих функции по таблице экспорта в динамической библиотеке.

Структура таблицы экспорта

```
DATA_DIRECTORY
        > DIRECTORY_ENTRY_EXPORT
        struct IMAGE_EXPORT_DIRECTORY
        {
            ...
            NumberOfFunctions
            ...
            AddressOfFunctions
            AddressOfNames
            AddressOfNameOrdinals
        }
```

Как и в случае с таблицей импорта, RVA таблицы экспорта можно узнать из директории данных (DATA_DIRECTORY).

В этом случае таблица экспорта - это первый элемент DATA_DIRECTORY. Значит RVA таблицы экспорта будет равен RVA для DATA_DIRECTORY. Данные здесь хранятся в соответствии со структурой IMAGE_EXPORT_DIRECTORY. Эта структура содержит несколько параметров, мы рассмотрим только те, которые для нас важны.

NumberOfFunctions - количество функций, которые экспортируются из библиотеки.

AddressOfFunctions - RVA, который указывает на массив RVA функций в библиотеке. Поле указывает на начало этого массива.

AddressOfNames - RVA, которое указывает на массив RVA имен функций в модуле. Поле указывает на начало этого массива.

Таким образом, если библиотека экспортирует 10 функций, массив, на который ссылается NumberOfFunctions, будет также состоять из 10 элементов, а поле NumberOfFunctions будет содержать значение 10. При этом, если все функции могут быть экспортированы по имени, то массив, на который ссылается AddressOfNames, тоже будет содержать 10 имен функций.

После того, как PE-загрузчик распарсил таблицу импорта, он знает список имен функций, которые ему нужны. Теперь, по таблице экспорта ему нужно получить RVA этих функций.

В таблице экспорта существует посредник между именами и адресами и называется это поле AddressOfNameOrdinals. Это поле указывает на массив, в котором количество элементов равно количеству элементов в массиве с именами функций (AddressOfNames). При этом каждому имени соответствует индекс из массива с адресами (AddressOfFunctions). Таким образом, используя этот массив, можно однозначно по имени функции получить RVA функции в библиотеке. Именно так PE-загрузчик и поступает.

Основные этапы загрузки PE-файла

Давайте перечислим основные этапы загрузки PE-файла:

1. Проверка PE-файла на корректность.

Сначала PE-загрузчик должен убедиться, что он имеет дело с PE-файлом. Это можно сделать по сигнатурам MZ и PE. Если сигнатуры на месте, значит идем дальше.

2. Поиск RVA PE-заголовка в DOS-заголовке.

Затем PE-загрузчик по смещению 0x3C получает RVA PE-заголовка и сразу переходит к нему.

3. Анализ PE-заголовка

PE-загрузчик анализирует PE-заголовок, получает все необходимые размеры и смещения, а также информацию по имеющимся секциям.

4. Копирование PE-файла в виртуальную память

На основе полученной информации PE-загрузчик размещает PE-заголовок и секции PE-файла в виртуальной памяти. PE-заголовок грузится по базовому адресу, а секции согласно указанному VirtualAddress и VirtualSize.

5. Анализ таблицы импорта

После того, как основной PE-файл загружен, PE-загрузчик начинает анализировать таблицу импорта. Ему необходимо получить список библиотек, которые необходимо подгрузить к основному PE-файлу, а также он получает список функций, для которых ему нужно будет найти RVA.

6. Копирование динамических библиотек в виртуальную память

На основе полученного списка библиотек, PE-загрузчик размещает динамические библиотеки в адресном пространстве процесса основного PE-файла. Как вы помните, динамические библиотеки тоже являются PE-файлами и принцип их загрузки почти не отличается от обычных EXE-файлов.

7. Анализ таблицы экспорта

После того, как динамические библиотеки загружены в память, PE-загрузчик выполняет поиск таблицы экспорта, анализирует ее и получает RVA функций, которые ему необходимы.

8. Коррекция адресов в основной программе

После того, как RVA всех компонентов, которые задействованы в программе известны, PE-загрузчик заменяет все RVA в основной программе на действительные адреса в виртуальной памяти.

9. Передача управления на точку входа

После того, как все подготовительные процедуры сделаны, PE-загрузчик передает управление на точку входа и программа начинает выполняться.